Building A Data Connector Code Generation Pipeline

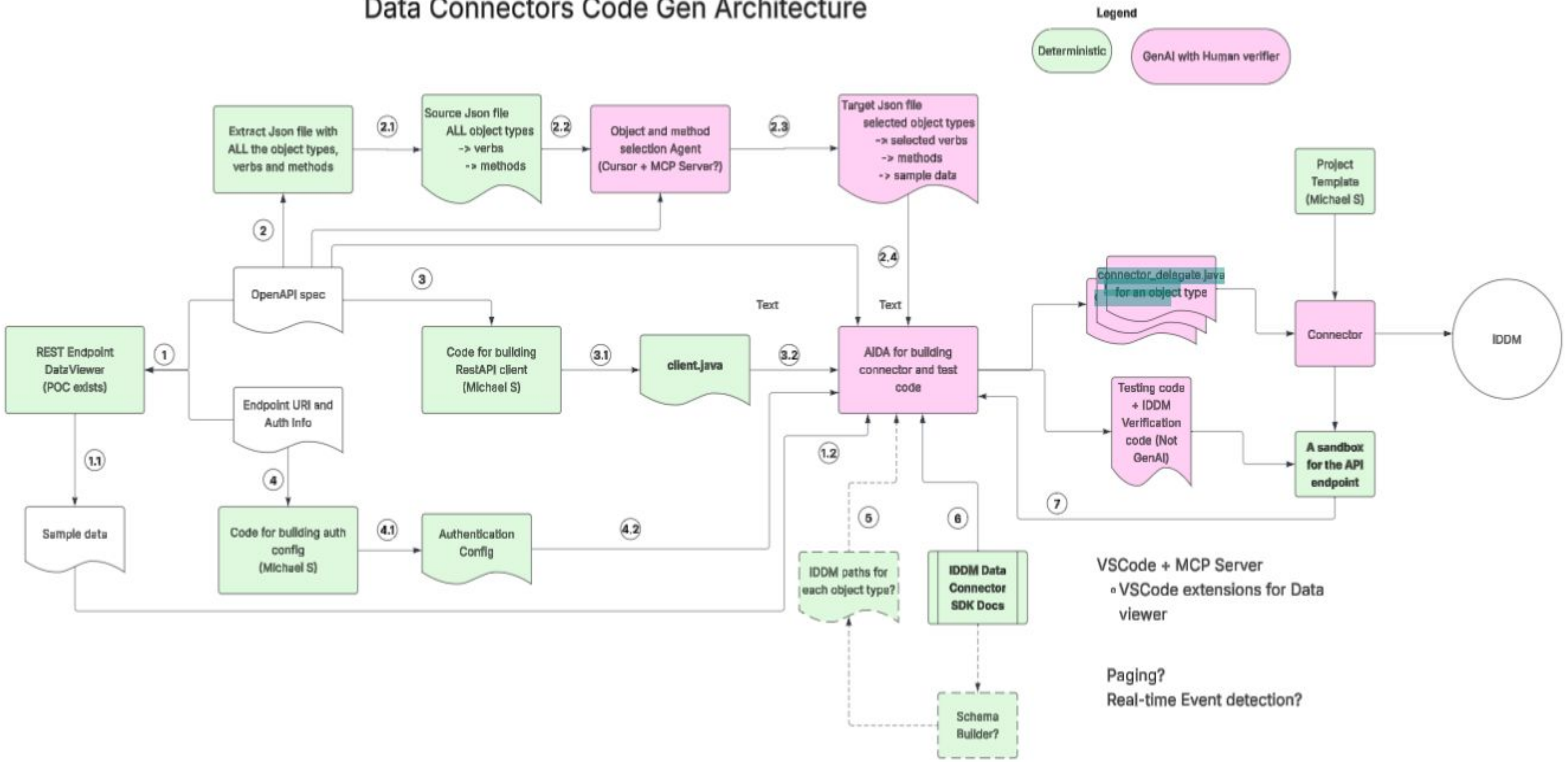
Krishna Saraiya

Introduction

- Automate generation of Data Connector Code Using DSPy
  - Parameters/Inputs
    - OpenAPI Spec
    - Radiant Logic IDDM SDK minimal user guide
    - Rest API Java Client Code (Api/Models files)
    - Target JSON file with selected objects/methods from OpenAPI Spec
- Model Context Protocol Tools (MCP) in Cursor
  - Use natural language commands to generate target json file, data connector code, run unit tests, etc.

Introduction

Data Connectors Code Gen Architecture


Legend
Deterministic
GenAI with Human verifier
Extract Json file with ALL the object types, verbs and methods
2.1
Source Json file ALL object types -> verbs -> methods
2.2
Object and method selection Agent (Cursor + MCP Server?)
2.3
Target Json file selected object types -> selected verbs -> methods -> sample data
2
2.4
Project Template (Michael S)
OpenAPI spec
3
Text
Text
connector_delegate.java for an object type
Connector
IDDM
REST Endpoint DataViewer (POC exists)
1
Code for building RestAPI client (Michael S)
3.1
client.java
3.2
AIDA for building connector and test code
Endpoint URI and Auth Info
Testing code + IDDM Verification code (Not GenAI)
A sandbox for the API endpoint
1.1
4
1.2
Sample data
Code for building auth config (Michael S)
4.1
Authentication Config
4.2
5
6
7
IDDM paths for each object type?
IDDM Data Connector SDK Docs
Schema Builder?

VSCode + MCP Server
- VSCode extensions for Data viewer

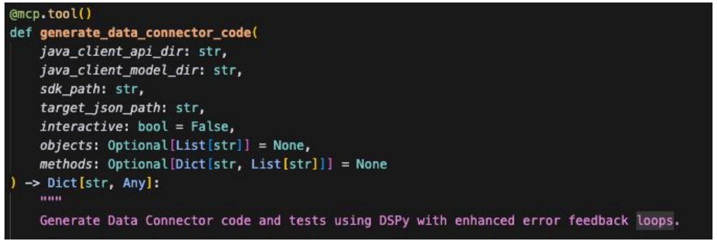
Paging?
Real-time Event detection?

Demo

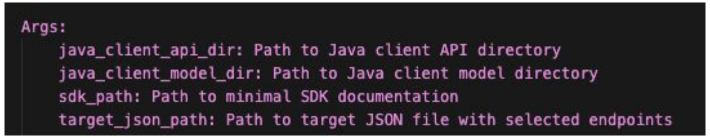
DSPy Signature Overview

- DSPy Chain Of Thought

1. Loads SDK Documentation
2. Looks through Java Client API Directory
   a. Provides API Client Examples for the LLM (specific to object)
3. Loads Target JSON Data
   a. Selected objects and methods from a Source JSON
   b. Creates Unified object structure for multi-object connectors
4. Object Analysis
   a. Determines selected objects to generate from Target JSON



```python
@mcp.tool()
def generate_data_connector_code(
    java_client_api_dir: str,
    java_client_model_dir: str,
    sdk_path: str,
    target_json_path: str,
    interactive: bool = False,
    objects: Optional[List[str]] = None,
    methods: Optional[Dict[str, List[str]]] = None
) -> Dict[str, Any]:
    """
    Generate Data Connector code and tests using DSPy with enhanced error feedback loops.
```



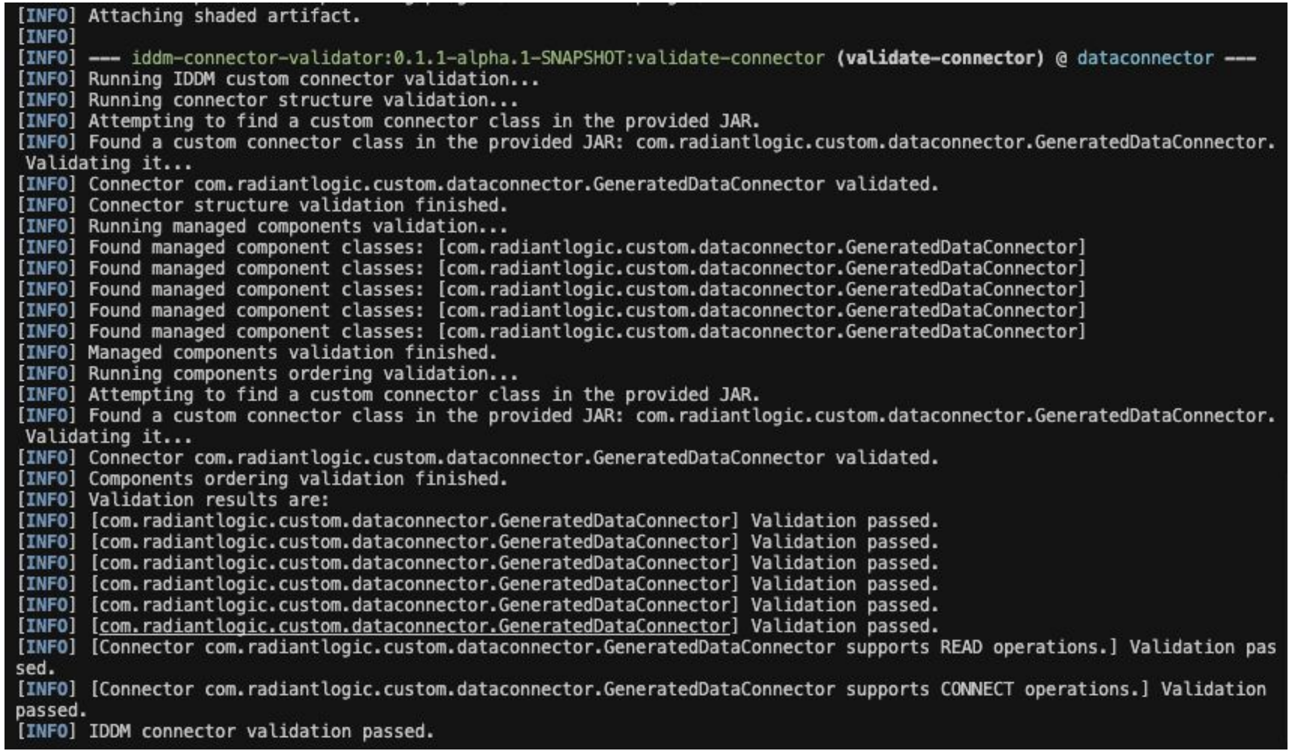
```
Args:
    java_client_api_dir: Path to Java client API directory
    java_client_model_dir: Path to Java client model directory
    sdk_path: Path to minimal SDK documentation
    target_json_path: Path to target JSON file with selected endpoints
```

Code Generation - What worked

- Basic Code Structure Generation
  - Proper package naming
  - Class Annotations
  - Interface Implementations
  - Some errors arise relating to build/compilation, but can either be fixed manually or through future error correction
- Test Generation
  - Tests for success and failure scenarios for searching objects
- Configuration File
  - JSON generated with proper metadata structure

Custom Connector Validation

Used to check correctness of the connector structure



```
[INFO] Attaching shaded artifact.
[INFO]
[INFO] --- iddm-connector-validator:0.1.1-alpha.1-SNAPSHOT:validate-connector (validate-connector) @ dataconnector ---
[INFO] Running IDDM custom connector validation...
[INFO] Running connector structure validation...
[INFO] Attempting to find a custom connector class in the provided JAR.
[INFO] Found a custom connector class in the provided JAR: com.radiantlogic.custom.dataconnector.GeneratedDataConnector.
 Validating it...
[INFO] Connector com.radiantlogic.custom.dataconnector.GeneratedDataConnector validated.
[INFO] Connector structure validation finished.
[INFO] Running managed components validation...
[INFO] Found managed component classes: [com.radiantlogic.custom.dataconnector.GeneratedDataConnector]
[INFO] Found managed component classes: [com.radiantlogic.custom.dataconnector.GeneratedDataConnector]
[INFO] Found managed component classes: [com.radiantlogic.custom.dataconnector.GeneratedDataConnector]
[INFO] Found managed component classes: [com.radiantlogic.custom.dataconnector.GeneratedDataConnector]
[INFO] Found managed component classes: [com.radiantlogic.custom.dataconnector.GeneratedDataConnector]
[INFO] Managed components validation finished.
[INFO] Running components ordering validation...
[INFO] Attempting to find a custom connector class in the provided JAR.
[INFO] Found a custom connector class in the provided JAR: com.radiantlogic.custom.dataconnector.GeneratedDataConnector.
 Validating it...
[INFO] Connector com.radiantlogic.custom.dataconnector.GeneratedDataConnector validated.
[INFO] Components ordering validation finished.
[INFO] Validation results are:
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [com.radiantlogic.custom.dataconnector.GeneratedDataConnector] Validation passed.
[INFO] [Connector com.radiantlogic.custom.dataconnector.GeneratedDataConnector supports READ operations.] Validation pas
sed.
[INFO] [Connector com.radiantlogic.custom.dataconnector.GeneratedDataConnector supports CONNECT operations.] Validation
passed.
[INFO] IDDM connector validation passed.
```

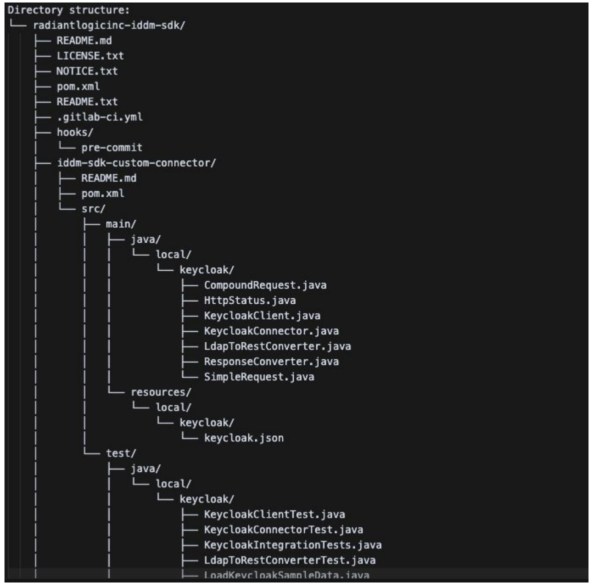
Code Generation - What did not work / Current Issues

- Examples Include
    - Build / Compilation Errors
        - Ideally should be fixed by LLM
    - Using the outdated Radiant Logic (radiantlogicinc-iddm-sdk.txt) file
        - Shared on June 22 but file was months older
        - Using older SDK docs yielded less than ideal data connector code
            - Also would not have worked with the validator plugin either way
    - Potentially Filename and Class Name Mismatches
    - Sometimes the LLM within cursor chat will hallucinate when given natural language commands
        - The MCP tool will sometimes read in wrong parameters/inputs
            - Ex. Wrong SDK doc file name or wrong paths to file

Context Engineering - What worked

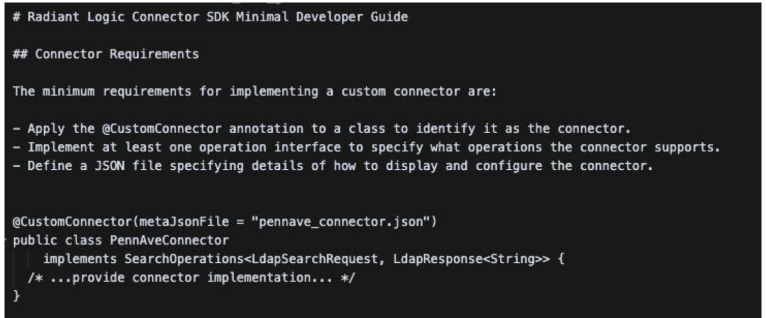
- Minimal SDK Documentation
  - Huge thanks to Michael Silva for providing the updated Minimal SDK
  - Reducing SDK context (around 247,000 tokens with Full SDK Doc) to 25,500 tokens (Minimal SDK)
    - Gemini 2.0 Flash-Lite context (around 1 M tokens)
    - Context overflow/slower generation no longer an issue

Original (and Outdated SDK)



```
Directory structure:
└── radiantlogicinc-iddm-sdk/
    ├── README.md
    ├── LICENSE.txt
    ├── NOTICE.txt
    ├── pom.xml
    ├── README.txt
    ├── .gitlab-ci.yml
    ├── hooks/
    │   └── pre-commit
    ├── iddm-sdk-custom-connector/
    │   ├── README.md
    │   ├── pom.xml
    │   └── src/
    │       ├── main/
    │       │   ├── java/
    │       │   │   └── local/
    │       │   │       └── keycloak/
    │       │   │           ├── CompoundRequest.java
    │       │   │           ├── HttpStatus.java
    │       │   │           ├── KeycloakClient.java
    │       │   │           ├── KeycloakConnector.java
    │       │   │           ├── LdapToRestConverter.java
    │       │   │           ├── ResponseConverter.java
    │       │   │           └── SimpleRequest.java
    │       │   └── resources/
    │       │       └── local/
    │       │           └── keycloak/
    │       │               └── keycloak.json
    │       └── test/
    │           ├── java/
    │           │   └── local/
    │           │       └── keycloak/
    │           │           ├── KeycloakClientTest.java
    │           │           ├── KeycloakConnectorTest.java
    │           │           ├── KeycloakIntegrationTests.java
    │           │           ├── LdapToRestConverterTest.java
    │           │           ├── LoadKeycloakSampleData.java
```

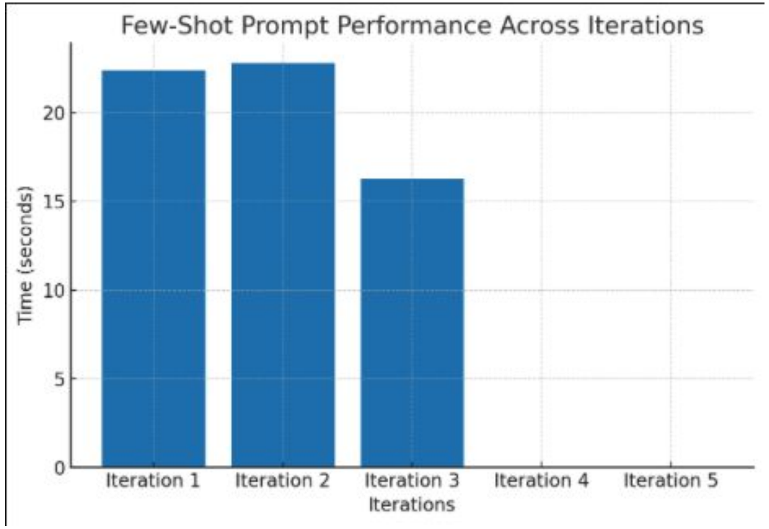
Updated Minimal SDK



```
# Radiant Logic Connector SDK Minimal Developer Guide

## Connector Requirements

The minimum requirements for implementing a custom connector are:

- Apply the @CustomConnector annotation to a class to identify it as the connector.
- Implement at least one operation interface to specify what operations the connector supports.
- Define a JSON file specifying details of how to display and configure the connector.


@CustomConnector(metaJsonFile = "pennave_connector.json")
public class PennAveConnector
    implements SearchOperations<LdapSearchRequest, LdapResponse<String>> {
  /* ...provide connector implementation... */
}
```

Context Engineering - What worked

- Structured Few-Shot Prompts
  - Giving example connectors from previous generation and configurations as templates improved some consistency



Error Type	Without Few-Shot	With Few-Shot	Reduction
Import errors	5+ per iteration	1–2 per iteration	60–80% reduction
Package errors	3+ per iteration	0–1 per iteration	70–100% reduction
Constructor errors	2+ per iteration	0–1 per iteration	50–100% reduction
Missing dependencies	4+ per iteration	1–2 per iteration	50–75% reduction

Feedback Driven Improvement

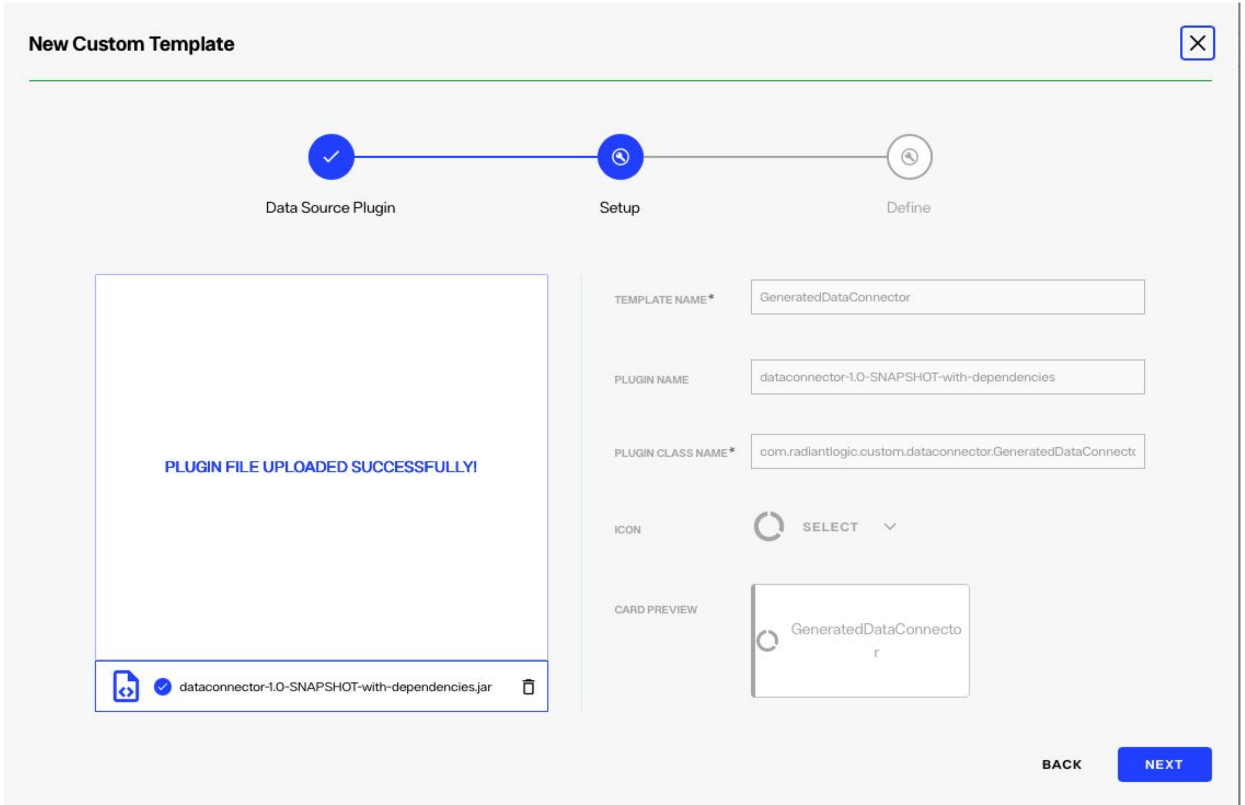
1. Iterative Refinement
    a. 5 iteration feedback loop
    b. Gives a score after each iteration (scoring system)
        i. Keeps best score
    c. Basic few-shot prompt examples hardcoded
    d. Not currently using LabeledFewShot Optimizer but will most likely use in the future

Example Refinement

Criteria	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Package name correctness	(0/15)	(15/15)	(15/15)	(15/15)	(15/15)
Config file reference	(0/15)	(0/15)	(0/15)	(15/15)	(15/15)
Class name correctness	(0/15)	(0/15)	(0/15)	(15/15)	(15/15)
Unified functionality	(0/15)	(0/15)	(15/15)	(15/15)	(15/15)
Compilation success	(0/20)	(0/20)	(0/20)	(0/20)	(20/20)
Package build	(0/20)	(0/20)	(0/20)	(0/20)	(20/20)
Test success	(0/10)	(0/10)	(0/10)	(0/10)	(10/10)
Total Score	**0/100**	**15/100**	**30/100**	**60/100**	**100/100**

Uploading to IDDM

- Within EOC Instance - Create Template - Custom Source Type



New Custom Template

Data Source Plugin | Setup | Define

PLUGIN FILE UPLOADED SUCCESSFULLY!

dataconnector-1.0-SNAPSHOT-with-dependencies.jar

TEMPLATE NAME*	GeneratedDataConnector
PLUGIN NAME	dataconnector-1.0-SNAPSHOT-with-dependencies
PLUGIN CLASS NAME*	com.radiantlogic.custom.dataconnector.GeneratedDataConnect
ICON	SELECT
CARD PREVIEW	GeneratedDataConnector

BACK NEXT

Future Improvements / To-Do

1. More robust error correction system
2. Test with other LLMs for comparison
   a. Mainly used Gemini 2.0 Flash Lite Preview
3. Currently testing with the smaller Okta Specs
   a. Ideally It needs to work with this
4. Working DSPy LabeledFewShot or other Optimizer

Thank you!

Questions?